Lecture 6:
Numerical linear algebra

Introduction

- Very often, one needs to solve (ex: regression)

$$\mathbf{Ax}=\mathbf{b}$$

A matrix
X unknown vector
b vector of scalars

Requirement

- The algorithm solving the problem should be
  - Efficient
  - Numerically stable

Linear regression models

Minimize

$$S(\beta_0, \beta_1, ..., \beta_p) = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_{1i} - ... - \beta_p x_{pi})^2$$

Solve the equation system $\frac{\partial S}{\partial \beta_0} = ... = \frac{\partial S}{\partial \beta_p} = 0$ that can be written $X^T X\beta = X^T Y$

where

$$X = \begin{pmatrix} 1 & x_{11} & . & . & x_{p1} \\ 1 & x_{12} & . & . & x_{p2} \\ . & . & . & . & . \\ . & . & . & . & . \\ 1 & x_{1n} & . & . & x_{pn} \end{pmatrix}$$

is a matrix of observed x-variables

Smoothing splines

- Minimize

$$S(\beta_0, \beta_1, ..., \beta_p) = \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int \{f''(x)\}^2 dx$$

- Solution is $f(x) = \sum_{i=1}^{n} N_j(x)\theta_j$

where θ is found from $\left(N^T N + \lambda \Omega_N\right)\theta = N^T Y$

Solving system of linear equations

- Therefore, it is important to solve **Ax=b**
- Aware of computer arithmetic! (recall a+x=x)

- Condition number
  - Original system $Ax = b$
  - Perturbed system $A\tilde{x} = \tilde{b} \qquad \tilde{x} = x + \delta x \qquad \tilde{b} = b + \delta b$

- Solution is good if small perturbation of *b* causes small perturbation of *x,* and since

$$\frac{\|\delta x\|}{\|x\|} \leq \|A\| \, \|A^{-1}\| \frac{\|\delta b\|}{\|b\|}$$

Solving system of linear equations

Condition number

$$\kappa(A) = \|A\| \, \|A^{-1}\|$$

Properties:

- Large condition number is a bad signal, but does not imply ill-conditioning
- If norm is L_2 then k is ratio of max.eigenvalue and min.eigenvalue
- Since $\kappa_2(A^{\mathrm{T}}A) = \kappa_2^2(A) \geq \kappa_2(A)$ problems in regression fitting may appear

Gaussian elimination and LU decomposition

Basic idea

- Consider the equation system $Ax = b$ where A is a square nonsingular matrix
- Set

$$C = \begin{pmatrix} a_{11} & a_{12} & . & . & a_{1p} & b_1 \\ a_{21} & a_{22} & . & . & a_{2p} & b_2 \\ . & & & & & \\ . & & & & & \\ . & & & & & \\ a_{p1} & a_{p2} & & & a_{pp} & b_p \end{pmatrix}$$

- Form

$$\begin{pmatrix} a_{11} & a_{12} & . & . & a_{1p} & b_1 \\ 0 & a_{22} - a_{12}a_{21}/a_{11} & . & . & a_{2p} - a_{1p}a_{21}/a_{11} & b_2 - b_1a_{21}/a_{11} \\ . & . & & & & \\ . & . & & & & \\ . & . & & & & \\ 0 & a_{p2} - a_{12}a_{p1}/a_{11} & & & a_{pp} - a_{1p}a_{p1}/a_{11} & b_p - b_1a_{p1}/a_{11} \end{pmatrix}$$

- and continue to eliminate variables one by one

Gaussian elimination and LU decomposition

- Two stages
  - Forward reduction
  - Backward substitution

- Forward reduction
  - Replacement $a_{j*}^{\mathrm{T}} x = b_j \quad \leftarrow \quad a_{j*}^{\mathrm{T}} x + c a_{k*}^{\mathrm{T}} x = b_j + c b_k$
  - Row interchange (if after k steps a_{kk} becomes zero)

$$a_{j*}^{\mathrm{T}} x = b_j \leftarrow a_{k*}^{\mathrm{T}} x = b_k$$
$$a_{k*}^{\mathrm{T}} x = b_k \leftarrow a_{j*}^{\mathrm{T}} x = b_j$$

Gaussian elimination and LU decomposition

- Forward reduction, equivalent elementary operations:
  - Replacement
    - Introduce E=I and replace e_{jk} by c (this is a lower triangular!)
  - Row interchange
    - Introduce E=I and interchange rows *j and k*
- !Elementary operations transform the system as

EAx=Eb

Gaussian elimination and LU decomposition

- Forward reduction, numerical issues:
  - Real $a_{kk}=0$, but computer a_{kk} became a small number -> interchange not happening
  - If dividing by a_{kk}, **Inf** may be obtained

- *Example*: 4 digits after comma

$$0.0001x_1 + x_2 = 1$$
$$x_1 + x_2 = 2$$

- -> partial pivoting, choose the equation that gives

$$\max_{i=k}^{n} |a_{ik}^{(k-1)}|$$

Gaussian elimination and LU decomposition

- Back substitution

$$x_n = \frac{\tilde{b}_n^{(n-1)}}{\tilde{a}_{nn}^{(n-1)}} \qquad x_{n-1} = \frac{\tilde{b}_{n-1}^{(n-2)} - \tilde{a}_{n-1,n}^{(n-2)} x_n}{\tilde{a}_{n-1,n-1}^{(n-2)}}$$

- Comments
  - Complexity $O(n^3)$
  - For all elementary operations, the inverse matrices are easily found

Gaussian elimination and LU decomposition

- After gaussian elimination (if no permutations),

$$U = E_n E_{n-1} \ldots E_1 A \Rightarrow A = \left(E_n E_{n-1} \ldots E_1\right)^{-1} U = LU$$

- If permutations are involved,

$$A = LUP$$

LU decomposition

Applications of LU

- Solving system of equations

- Inverse matrix

$$A^{-1} = U^{-1} L^{-1}.$$

- Determinant

det(A)=det(L)*det(U)=product of diagonal el-s

QR decomposition

- A can be decomposed

$$A = QR$$

  - Q orthogonal
  - R upper triangular (trapezoidal)

- If A=m x n , m>n, $R = \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$

QR factorization

- Use of QR in regression (see proof…)

$$\hat{b} = \left(X^T X\right)^{-1} X^T y = R^{-1} Q^T y$$

- R is upper triangular -> inverse by back substitution (fast)

How to do QR

- Gram-Schmidt orthonormalization

$$\operatorname{proj}_{\mathbf{e}}\mathbf{a} = \frac{\langle \mathbf{e}, \mathbf{a} \rangle}{\langle \mathbf{e}, \mathbf{e} \rangle}\mathbf{e}$$

$$\begin{aligned}
\mathbf{u}_1 &= \mathbf{a}_1, & \mathbf{e}_1 &= \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} \\
\mathbf{u}_2 &= \mathbf{a}_2 - \operatorname{proj}_{\mathbf{e}_1} \mathbf{a}_2, & \mathbf{e}_2 &= \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} \\
\mathbf{u}_3 &= \mathbf{a}_3 - \operatorname{proj}_{\mathbf{e}_1} \mathbf{a}_3 - \operatorname{proj}_{\mathbf{e}_2} \mathbf{a}_3, & \mathbf{e}_3 &= \frac{\mathbf{u}_3}{\|\mathbf{u}_3\|} \\
&\vdots & &\vdots \\
\mathbf{u}_k &= \mathbf{a}_k - \sum_{j=1}^{k-1} \operatorname{proj}_{\mathbf{e}_j} \mathbf{a}_k, & \mathbf{e}_k &= \frac{\mathbf{u}_k}{\|\mathbf{u}_k\|}
\end{aligned}$$

$$\langle \mathbf{e}_i, \mathbf{a}_i \rangle = \|\mathbf{u}_i\|$$

How to do QR

$$\mathbf{a}_1 = \langle \mathbf{e}_1, \mathbf{a}_1 \rangle \mathbf{e}_1$$
$$\mathbf{a}_2 = \langle \mathbf{e}_1, \mathbf{a}_2 \rangle \mathbf{e}_1 + \langle \mathbf{e}_2, \mathbf{a}_2 \rangle \mathbf{e}_2$$
$$\mathbf{a}_3 = \langle \mathbf{e}_1, \mathbf{a}_3 \rangle \mathbf{e}_1 + \langle \mathbf{e}_2, \mathbf{a}_3 \rangle \mathbf{e}_2 + \langle \mathbf{e}_3, \mathbf{a}_3 \rangle \mathbf{e}_3$$
$$\vdots$$
$$\mathbf{a}_k = \sum_{j=1}^{k} \langle \mathbf{e}_j, \mathbf{a}_k \rangle \mathbf{e}_j$$

→ **A=QR,** Q-orthogonal

$$Q = [\mathbf{e}_1, \cdots, \mathbf{e}_n] \qquad \text{and} \qquad R = \begin{pmatrix} \langle \mathbf{e}_1, \mathbf{a}_1 \rangle & \langle \mathbf{e}_1, \mathbf{a}_2 \rangle & \langle \mathbf{e}_1, \mathbf{a}_3 \rangle & \dots \\ 0 & \langle \mathbf{e}_2, \mathbf{a}_2 \rangle & \langle \mathbf{e}_2, \mathbf{a}_3 \rangle & \dots \\ 0 & 0 & \langle \mathbf{e}_3, \mathbf{a}_3 \rangle & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

Cholesky decomposition

- For symmetric and positive definite A,

$$\mathbf{A=T^TT}$$

  - **T** upper triangular
  - T is sometimes called square root
  - There is a simple algorithm (see book)
  - Cholesky decomp. algorithm is numerically stable
  - Complexity $O(n^3)$
  - Only one matrix with n(n+1)/2 elements needed for decomp.

Cholesky decomposition

Applications:

- Solving **Ax=b** (see how I goes…)
- Nonlinear optimization (quasi-newton methods, next lecture)
- Generating correlated variables:

Suppose we need to generate $N(\mu,\Sigma)$:

1. Take i.i.d. N(0,1) sequence $X=(X_1,\ldots X_n)$
2. Compute Cholesky factor or matrix square root, i.e. matrix A: $AA^T=\Sigma$
3. Compute Y as $\mu+AX$

Observe: $EY=\mu$, $cov(Y)=AA^T$

Other decompositions

Factorization	Restrictions	Properties of Factors
SVD, page 28 $A_{nm} = U_{nn} D_{nm} V_{mm}^{\mathrm{T}}$	none	U orthogonal V orthogonal D nonnegative diagonal
variations:	for symmetric A, $A = VCV^{\mathrm{T}}$	
LU, page 215 $A_{nn} = L_{nn} U_{nn}$	A square, (others)	L full-rank lower triangular U upper triangular
variations:	with partial pivoting, $A = LUP$ with full pivoting, $P_1 A P_2 = LU$ $A = LDU$, with D diagonal and $u_{ii} = 1$	
QR, page 216 $A_{nm} = Q_{nn} R_{nm}$	none	Q orthogonal R upper triangular
variations:	skinny QR for $n > m$, $A = Q_1 R_1$	
Cholesky, page 216 $A_{nn} = L_{nn} U_{nn}$	A nonnegative definite	L full-rank lower triangular U upper triangular
diagonal, page 27 $A_{nn} = V_{nn} C_{nn} V_{nn}^{\mathrm{T}}$	A symmetric	V orthogonal C diagonal
square root, page 29 $A_{nn} = (A_{nn}^{\frac{1}{2}})^2$	A nonnegative definite	$A_{nn}^{\frac{1}{2}}$ nonnegative definite

Conjugate gradient method

- Solving **Ax=b** is equivalent to minimizing

$$f(x) = \frac{1}{2}x^{\mathrm{T}}Ax - x^{\mathrm{T}}b.$$

General scheme for iterative methods:

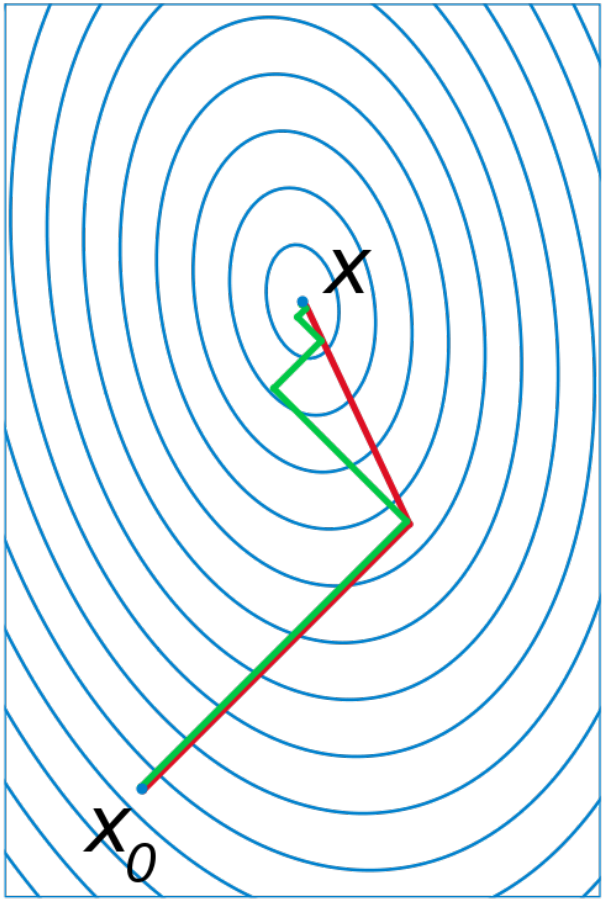
1. Choose starting point $x^{(0)}$, i=1
2. Choose direction $p^{(i)}$ and scalar $\alpha^{(i)}$
3. Update $x^{(i)} = x^{(i-1)} + \alpha^{(i)} p^{(i)}$
4. Until convergence

Conjugate gradient method

- Directions are conjugate if

$$(p^{(k)})^{\mathrm{T}} A p^{(i)} = 0, \quad \text{for } i = 1, \ldots, k-1$$

- Why do we need CG method?



Conjugate gradient method

- Finding of conjugate directions is done directly in the algorithm:

0. Input stopping criteria, ϵ and $k_{\max}$.
   Set $k = 0$; $r^{(k)} = b - Ax^{(k)}$; $s^{(k)} = Ar^{(k)}$; $p^{(k)} = s^{(k)}$; and $\gamma^{(k)} = \|s^{(k)}\|^2$
1. If $\gamma^{(k)} \leq \epsilon$, set $x = x^{(k)}$ and terminate.
2. Set $q^{(k)} = Ap^{(k)}$.
3. Set $\alpha^{(k)} = \frac{\gamma^{(k)}}{\|q^{(k)}\|^2}$.
4. Set $x^{(k+1)} = x^{(k)} + \alpha^{(k)} p^{(k)}$.
5. Set $r^{(k+1)} = r^{(k)} - \alpha^{(k)} q^{(k)}$.
6. Set $s^{(k+1)} = Ar^{(k+1)}$.
7. Set $\gamma^{(k+1)} = \|s^{(k+1)}\|^2$.
8. Set $p^{(k+1)} = s^{(k+1)} + \frac{\gamma^{(k+1)}}{\gamma^{(k)}} p^{(k)}$.
9. If $k < k_{\max}$,
   set $k = k + 1$ and go to step 1;
   otherwise
   issue message that
   "algorithm did not converge in $k_{\max}$ iterations".

Conjugate gradient method

Comments

- Most appropriate for large (sparse) A, nxm>=1000x1000
- Converges in *n* iterations
- In computer arithmetic, may take more time

Reading

- Chapter 5
- Wikipedia